Space-Cube: A Flexible Computer Architecture Based On Stacked Modu es

Gary Bolotin

Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109
bolotin@telerobotics.jpl.nasa.gov

Abstract

This paper presents a flexible, computer architecture based on stacked interchangeable modules. The architecture is ideally suited to implementation using stacked multichip modules (MCMs). The architecture, by making use of all sides of a module stack, allows simple, single bussed modules, together with a gate-way module, to be easily configured into a variety of more complicated architectures.

Introduction

Future space computing systems need low volume, mass, and power electronics.[1] Through the use of innovative architectural, and microelectronic packaging design this goal can be achieved. These architectures need to:

1). easily incorporate a variety of different requirements.
2). provide a framework by which data bus protocol and definition, module package size, interconnection and shape, can be standardized in order to maintain modularity and architectural flexibility.
3). create an environment that encourages module inheritance from mission to mission.
4). be independent of processor type.
5). incorporate new technologies as they become developed.

The Space-Cube Architecture addresses these issues.

Comparison With State of the Art

Prior to the Space-Cube, multiprocessor and distributed systems were made up of boards or modules plugged, side by side, in a card cage or frame. The modules were then connected by means of a single backplane or bus along the rear surface of the modules. Unfortunately, this single bus strategy creates a system bottle neck. Multiple busses, when required, could be created, but the lack of an underlying framework for multiple bus construction, results in the possibility of a different design for every application. The Space-Cube provides this necessary framework.

Backplane interconnections make use of only one surface of a stack of boards. Multiple bus architectures implemented in a backplane would require either crossing busses or nonidentical modules. Crossing buses limit the scalability of a design, while nonidentical modules increase the cost of a design. Space-Cube architecture implementations, by making use of all the surfaces of the modules, are implemented with noncrossing, module to module connections, and identical module construction. This eliminates the scalability restriction and reduces cost at the same time.

Stacked MCM Technology

The Space-cube architecture is ideally suited to implementation using stacked multichip modules (MCMs). Figure 1 shows an 3-D MCM stack implemented using elastomaric elements for the vertical interconnections. This MCM stacking technique is suitable for a spaceboure computer, and is being proposed or used in flight computers of Space Computer Corporation, NASA Jet Propulsion Laboratory and others.[1,2,3]

This module stacking technique replaces the traditional backplane with a 3-D interconnect. In this case an elastomeric interconnect. Elastomeric connections are made out of a compressable matireial that conducts only in the vertical direction. The stack shown is held together by 4 bolts, one in each corner. These bolts hold the 3-D interconnect elements, under compression, agaist pads on along the perimter of each substrate. The elastomeric connection allows for serperation of the individual elements of the stack for easy repair, replacement or reconfiguration.

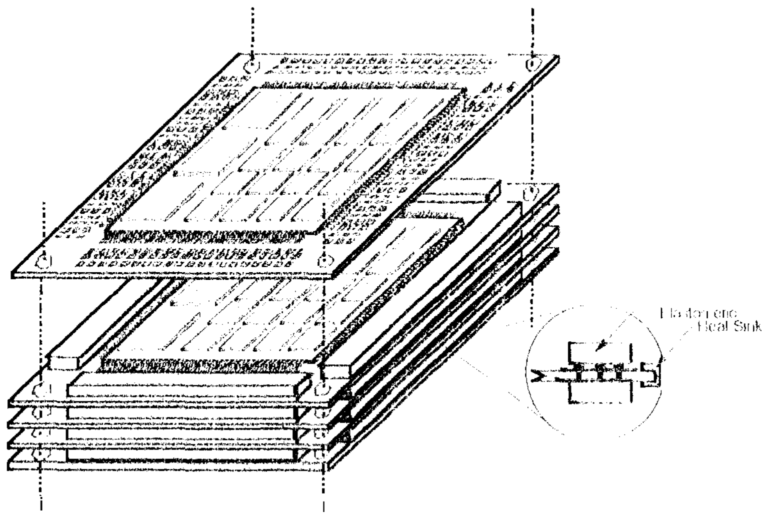

Figure 1. 3-D MCM Stack

The Space-Cube Architecture

The Space-Cube architecture is a flexible, processor independent, computer architecture based on stacked interchangeable modules. The modules are regular polygon (i.e. square, hexagon, etc.) in shape. These modules when stacked on top of each other, are easily configured to a variety of more complicated architectures. The resulting system is easily expanded and maintained.

The Module Building Block

The basic building block of this architecture is called a module, as illustrated in figure 1. This module represents an abstraction of a single MCM.

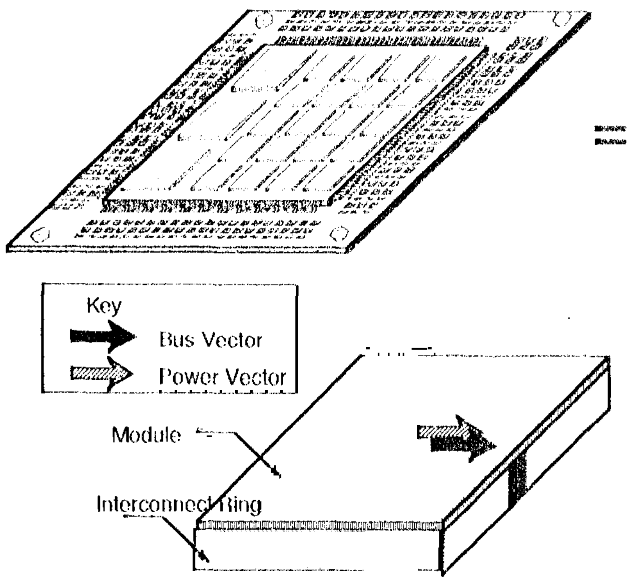

Figure 2. The basic module

The module shown is in the shape of a square, and subsequently can be placed in any of four possible orientations. (N, S, E, W) The communication between this modules through a high bandwidth bus, represented by the arrow/vector. A single bus is confined to fit within one side of a module. A module that interfaces to more then one bus would have more then one arrow.

Even though a module may only require a single bus, Space-Cube modules are required to connect the I/O pads of remaining sides through from one surface to another. This allows the communication occuring on the other surfaces to be uninterupted by this module.

Vectors

To simplify the discussion vectors are used. Vectors can be used to represent properites of modules. Five vector types will be discussed; bus vector; power vector; thermal vector; testability vector; and I/O vector. These vectors represent independant features of a module and can be directed independantly of each other. However, for puposes of this paper only the bus and power are to be discussed. They are to be combined into one vector represented by a hevy black arrow.

The bus vector represents the high bandwidth bus used for basic communictaion between modules. A single vector represents a unique bus. Modules that need to communicate with more then one bus will have more then one vector.

The power vector represents the direction by which a module recieves its power. This vector may or may not be independant of the bus vector.

The I/O vector represents the direction by which a module communicates with the outside world. Bus vectors and I/O vectors can not point in the same direction if I/O connections would interfer with module to module connections on that side of the module.

The thermal vector represents the dominant direction of heat conduction. The thermal vector can be used for analizing the thermal properties of a stack.

The testability vector represent the direction in which a module would recieve the five I.E.E.E. 1149.1 module scan test signals. The use of these signal would allow in system test and debugging.

Interconnect Rings

Interconnections between modules are done by means of interconnect rings, as shown in figure 3. The interconnect ring represents an abstraction of the elastomeric connections shown in figure 1. The interconnect ring provides the electrical, thermal, and mechanical interface between the modules of the system. An interconnect ring is a hollow four sided structure which can carry up to four buses from module to module. Any side of the interconnect ring can (not) independently conduct the bus to its neighboring module depending on whether it is conductive or nonconductive. This would represent a condutive or noncondutive elastomeric. This connection is abstracted by a thick black line or lack there of. Figure 3 shows all possible ring flavors in a four sided system. These rings, just like the modules, can be placed in any of four possible orientations. (N, S, E, W)

A module stack is constructed by stacking modules together, leggo-style, with interconnect rings as shown in Figure 4. This results in a physically and mechanically robust system.

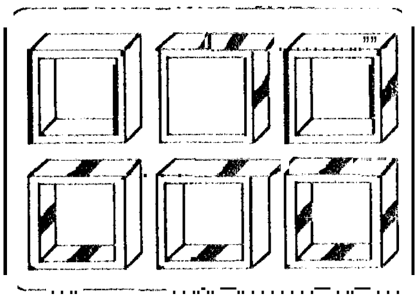

Figure 3. Interconnect Rings

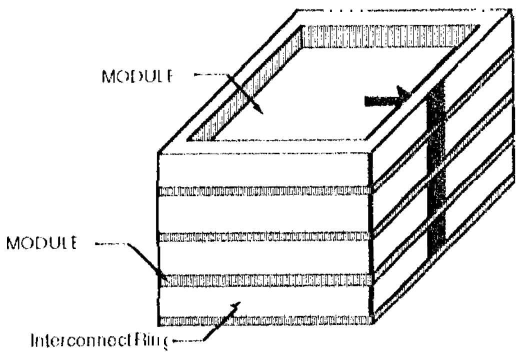

Figure 4. Module Stack

A Space-Cube Based Computer System

To illustrate how the Space-Cube Architecture can be used to construct a computer system, we first need to construct several types, "flavors" of modules. The granularity of the modules is to be chosen based on the problem and technology at hand. The granularity must be small enough so that there is sufficient commonality among systems to justify this approach. The granularity must also be chosen large enough to overcome the overhead of this bus based architecture.

Module Flavors

For purposes of this proposal we will discuss four possible module flavors:

1). CPU
2). Memory
3). Gateway
4). I/O

These modules types will be discussed in detail and are illustrated in figure 5. Systems can then be created by linking the desired modules into a stacked design.

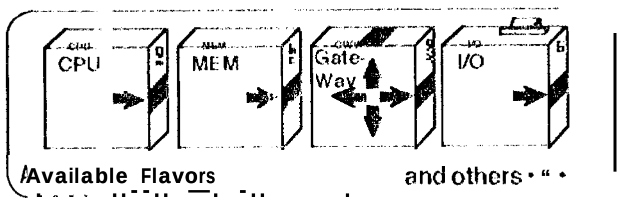

Figure 5. Module Flavors

The CPU flavor is a self contained processing module. This module contains a microprocessor along with its local bus and a single interface on to the Space-Cube bus: Space-bus. The CPU will access the Space-bus when it is performing a read or write of external data. The remainder of the time the Space-bus is free to be used by other modules.

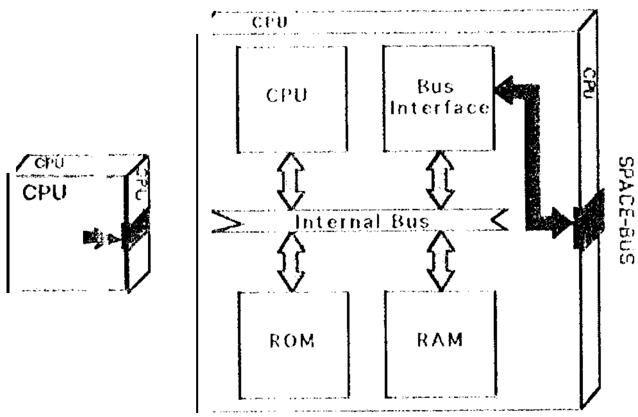

Figure 6 CPU

The memory flavor is a single ported memory unit mapped onto the Space-Cube bus. This unit represents the mass storage of a Space-Cube stack. The memory module functions as a slave module and will only respond when accessed by the current bus master on the Space-bus.

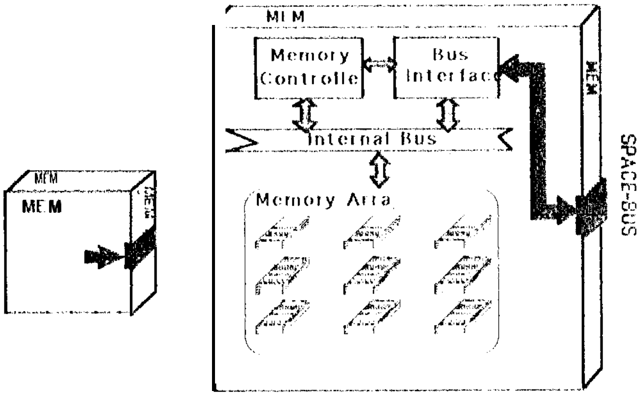

Figure 7. Memory

The I/O flavor is a single ported unit and presents a means of getting data into and out of Space-Cube stack. The I/O module functions as a slave module and will only respond when accessed. The I/O module has two external connections. The first is the connection to the Space-bus and is represeneted by a bus vector. The second connection is from the I/O module to the external enviorment. This connection is represented by an I/O vector.

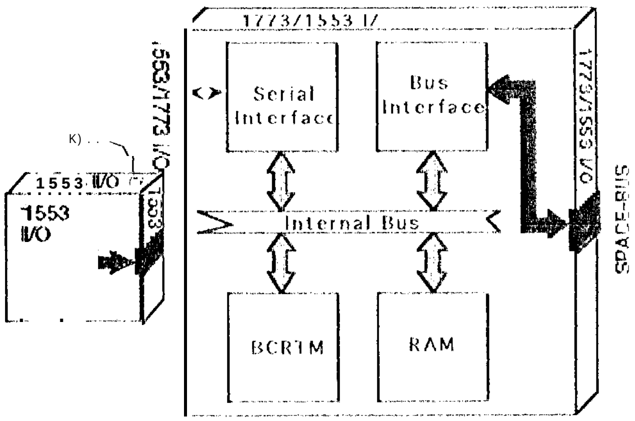

Figure 8. I/O

The gate-way flavor is a multi-ported unit and is the only module that is unique to the Space-Cube architecture. The module is basicly a four ported crossbr network together with the arbitration logic necessary to make the module work. The gate-way provides a means of connecting one bus surface onto any one of the three other surfaces. In a four sided system gateways come in two varietys, four and eight ported. The four ported gate-way treats each side of the stack as a single unique bus. A four ported gate-way module when inserted into a stack would not represent a break in the bus structure. The eight ported varity seperates busses above the module from those below. This results in eight unique busses.

The four ported gate-way flavor is illustrated in figure 9. Since only two concurrent connections are possible, this module requires only two internal busses. For example, if port A is talking to port D, then the only possible remaining connection is from port B to port C. If a request were to come in on a port for a port that is occupied the request will be denied.

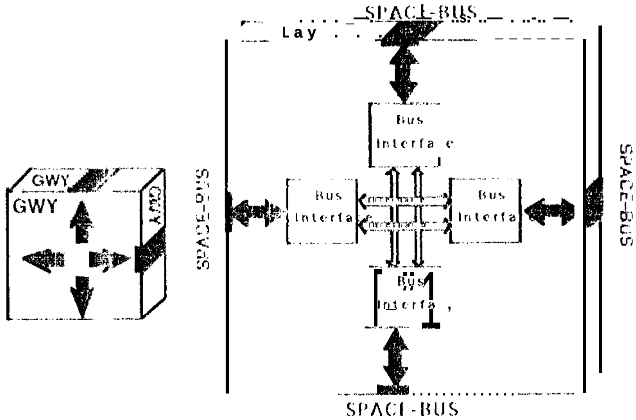

Figure 9. The Four-Ported Gate-way

The eight ported gate-way flavor is illustrated in the following figure. The eight ported gate-way, requires only four internal busses. This variety of gate-way acts as a seperator of the busses above and below the module.

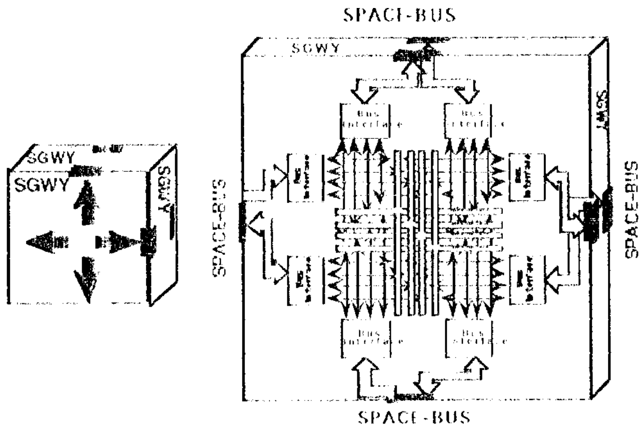

Figure 10. The Eight-Ported Gate-way

Examples

Module stacks are constructed by simply stacking modules together with interconnect rings. This section describes examples of some Space-Cube systems.

Single Processor

A simple example of a Space-Cube is a simple single processor with its own memory and I/O. A single processor is constructed by stacking a CPU module together with a memory and I/O module as shown. Since the three modules are all stacked in the same direction, the three modules share the same bus.

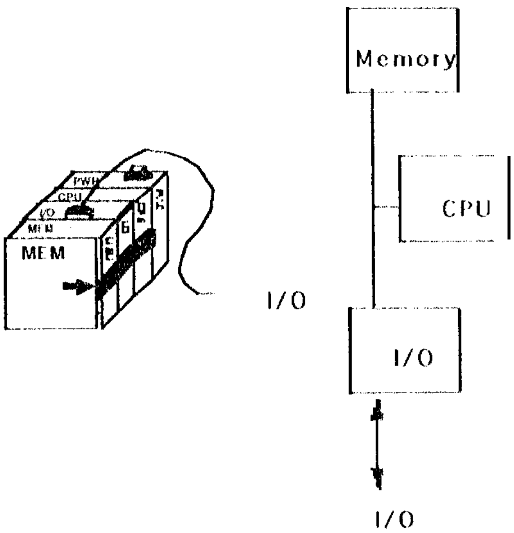

Figure 11. A Single Processor

Dual Processor

A simple dual processor, with a common shared dual-ported memory can be constructed by putting the processors, and the shared memory module, at three unique angles, together with a gate-way. The local bus for each processor is assigned a unique surface. The shared memory bus is given a third. The gate-way module which is placed between the processors and the shared memory completes the stack

This example implenmentation was chosen because it illustrates some of the key advantages of the Space-Cube. Under fault-free conditions the system results in a distributed processing environment with one processor dedicated to science data processing, and the other to engineering data processing. However, upon the detection of a fault in either processor, the surviving processor will assume the responsibilities of the other. Fault tolerance is made possible because of the cross-strapping between the processors, memories and I/Os.

The mass and power of this example architecture is comparable to a single string option yet it can provide similar fault protection compared to a true dual string architecture. Only the critical I/O hardware is duplicated.[4]

This architecture can easily be implemented in a Space-Cube. The following module flavors are needed:

1). Processor
2). Memory
3). Gateway
4). I/O

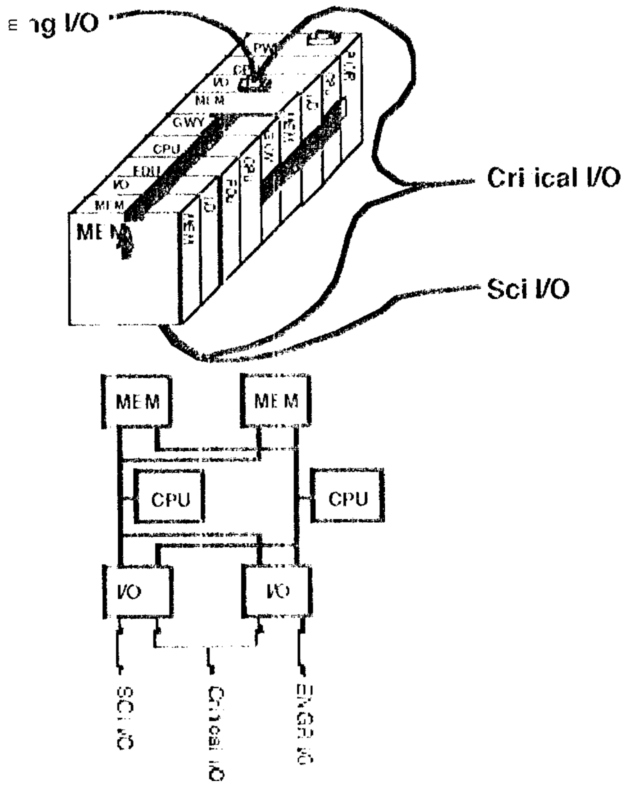

Figure 12. A Dual Processor With Shared Memory

The gate-way module provides the means for communiction between surfaces of the stack, thus making complicated archilecures possible. Figures 13 and 14 show the gate-way in action. Since each processor is assigned its own unique surface, or bus, communcition between a processor and its own local memory can be achieved with out interfering with the other processor. A potential conflict can occur only when two or more processors try to access the same shared bus.

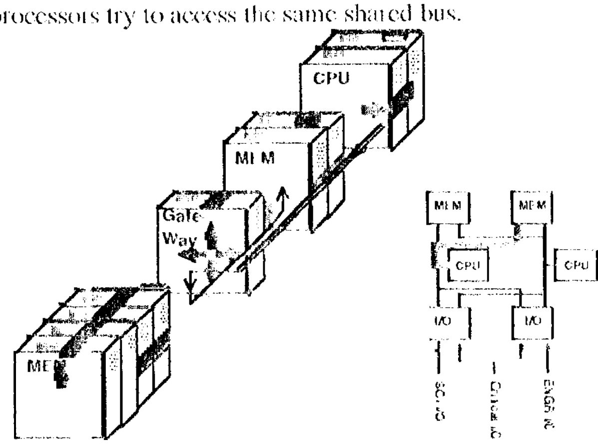

Figure 13. The Gate-way In Action

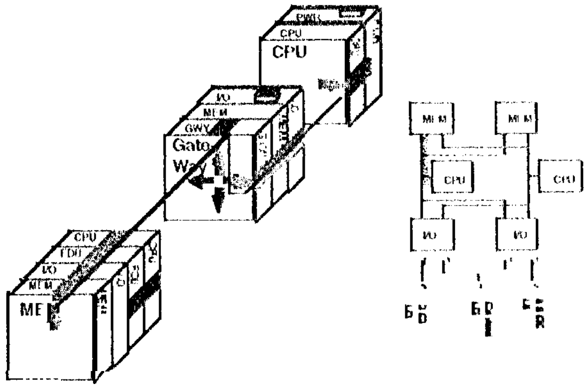

Figure 14. The Gate-way In Action Cont.

A multiprocessor can be constructed by stacking four processors in each of the four directions, both above and below a gate-way module. Each processor has its own unique bus. Comunication between the processors of a cluster is done by means of the gate-way module.

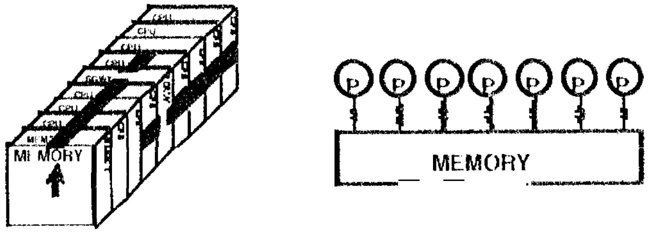

Figure I .5.1'1 (x"('s Sol Cluster

Conclusion

The Space- Cube architecture is computer achitecture based on ?-I) stacked modules. The architecture allows simple single bussed modules to form a wide varity of complicated architectures, by making use of all sides of the module stack. Module inheiratence, is also enchouraged.

The Space-Cube architecture is not a computer architecture in and of itself, by provides a means of mapping existing computer architectures into a 3-D stack of moudules in an elegant fashion. All connections are done with simple module to module connections, and all with out any crossing wires.

Work is underway to demonstrate the architecture using comercial of the shelf components. PC104 modules together with PCMCIA cards are being used as basic module building bocks. The PC104 cards are adapted to a Space-Cube module by attaching connectors to all four sides of a PC104 card. The design of a gate-way module, the only module that needs to be designed from scratch has also been started.

Although idealy suited for stacked MCM technology the Space-Cube architecture is technology independant. The architecuture can be demonstrated with comercial of the shelf components today, and provide a framework for stacked die implementations in the future.[5].

Acknowledgment

This research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

[1] Leon Alkalaj, MESUR Network Integrated Microelectronics Study, Jet Propulsion Laboratory, California Institute of Technology, D-11192, January 7, 1994.

[2] Leon Alkalaj, The Design and Implementation of NASA's Advanced Flight Computing Module, IEEE Multichip Module Conference, Santa Cruz, CA, January 31-February 2, 1995.

[3] N. J. Tenecketges, W. J. Jacobi and L. A. Wadsworth, A High-Performance MCM-BasedSpaceborne Processor, ICMCM Proceedings '92

[4] Savio Chau, An Optimal Architecture of Pluto Fast Flyby, Jet Propulsion Laboratory Internal Document, October 16, 1994.

[5] Military and Defense Electronics, ISC Develops Mix and Match 3-D Package For AF, Vol. 5, No. 5 April 1994.